REAL-TIME MONITORING SYSTEM FOR GEOTHERMAL WELL FLOW TEST USING INTERNET of THINGS

Ende Cecep Ependi[1], Reza Diharja[2], Nurdina Widanti[3]

[1,2,3]Program Studi Teknik Elektro, Fakultas Teknologi Industri, Universitas Jayabaya, Indonesia
E-mail: cecep.ependi@hotmail.com, Diharja@gmail.com, dina.dinawidi.widi7@gmail.com

Abstract—*Geothermal energy is a renewable resource with significant potential for sustainable utilization. Accurate and real-time monitoring of parameters during geothermal welltesting is essential to ensure system performance and anticipate potential disturbances. This study aims to design and develop a geothermal well flow monitoring device based on Arduino Uno and ESP8266 microcontrollers using the Message Queuing Telemetry Transport (MQTT) communication protocol. The system is designed to read industry-standard 4–20 mA sensor signals representing field parameters, wirelessly transmit data to an MQTT broker, and display it on a web-based interface. Laboratory-scale testing was conducted using a 4–20 mA signal generator to simulate field sensor outputs. Evaluation included reading accuracy, data transmission stability, and system response time. Test results showed the device can read signals with a maximum deviation of ±1.08% from the reference value, transmit data with a 99% success rate, and achieve an average latency of 0.8 seconds. In conclusion, the geothermal monitoring device based on ESP8266 and MQTT protocol developed in this study demonstrates reliable performance at the laboratory scale and holds potential for field implementation in industrial-scale monitoring.*

Index Terms— *Geothermal Energy, Welltesting, MQTT, Arduino Uno, ESP8266, Real-Time Monitoring, 4–20 mA Sensor.*

I. Introduction

Well testing is an essential stage in geothermal well development, conducted after drilling reaches the targeted depth and the production zone is successfully penetrated. This stage aims to determine the well's production capacity and fluid characteristics, which serve as the basis for defining the optimal operating conditions. Mass flow rate and fluid enthalpy data are crucial for estimating well potential at various wellhead pressures [1], [2]. During the well test process, the produced fluid is directed to a separator to split brine and steam, followed by the measurement of key parameters such as pressure, temperature, and flow rate for each phase.

Traditional monitoring methods for well testing often rely on manual recording at predefined time intervals. This approach presents several limitations, including delays in detecting changes in conditions, inaccurate records, and potential data loss due to weather disturbances or operational constraints. The manual process also requires additional time for data entry into a computer and subsequent processing before the results can be analyzed by the engineering team.

With the advancement of Internet of Things (IoT) technology, microcontroller-based data acquisition systems have emerged as a promising solution to improve monitoring efficiency and accuracy. This study designs and implements a data acquisition system using an Arduino microcontroller and an SD card–based datalogger shield directly connected to transmitter sensors. The system is capable of continuously recording vital operational parameters, such as wellhead pressure, fluid temperature, fluid pressure at the lip pipe, and water flow rate.

In addition to local storage for ensuring data integrity, the system also transmits real-time data via a NodeMCU ESP8266 module using the MQTT protocol to the Adafruit IO platform. The transmitted data are then visualized on a web-based dashboard to support remote monitoring and faster decision-making. This implementation is expected to enhance efficiency, accuracy, and timeliness in geothermal well testing while supporting the digital transformation of operational data management.

II. Literature Review

Research on Internet of Things (IoT)–based monitoring systems has been widely conducted, particularly in the energy and environmental sectors. Rangan et al. [4] developed an IoT-based monitoring system for air temperature and humidity in the XYZ chemical laboratory using a DHT11 sensor and a NodeMCU ESP8266 as the data processor, enabling DHT11 sensor readings to be displayed on a website. Hafidhin et al. [5] utilized an Arduino UNO to control an automatic fish-drying process based on temperature and humidity sensor readings.

The use of analog sensors with a 4–20 mA current output is common in industrial environments, including geothermal operations. To interface such sensors with a microcontroller, the signal must first be converted into a

voltage using a conversion module such as the HW-685. Subsequently, a high-resolution ADC such as the ADS1115 is employed to precisely measure the voltage. Risky Martin [6] developed a prototype voltmeter using the ADS1115 and Arduino UNO for high-accuracy DC resistivity measurements.

For long-distance data transmission, the ESP8266 module is widely used due to its Wi-Fi connectivity and compatibility with the MQTT protocol. Manullang et al. [7] implemented the ESP8266 in an IoT-based motorcycle security system with internet-based notifications. Oklilas et al. [8] also built a temperature monitoring system using the ESP8266 and MQTT protocol to transmit data to a server and display it in real time.

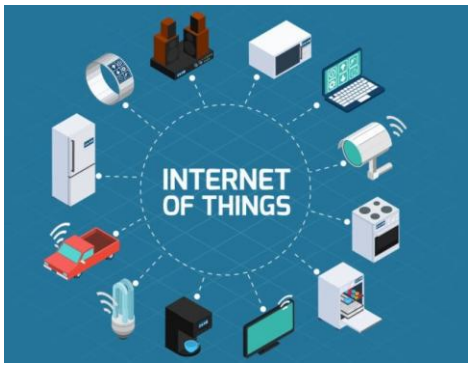
Pahlevi, Sukarno, and Erfianto [3] demonstrated that the MQTT protocol is highly suitable for IoT systems due to its lightweight and efficient nature, particularly for periodic sensor data transmission. Furthermore, web-based monitoring dashboards have been developed in various studies, such as Nurhadi A. and Ricki Sastra [9], who created a desktop application for employee leave requests through a web interface.

All components used in this study—from pressure and temperature sensors, current-to-voltage conversion modules, ADC measurement via ADS1115, Arduino microcontroller, ESP8266 Wi-Fi module, to the MQTT-based dashboard—have been implemented in similar monitoring systems in various fields. This research integrates these components to develop a geothermal well test monitoring system accessible in real time from locations outside the field site.

Several studies have incorporated microSD card technology for local data storage in embedded systems. For instance, Pranoto [15] developed a prototype RFID-based attendance system utilizing a microSD card to locally store attendance records, demonstrating the effectiveness of microSD cards for reliable and portable data logging in microcontroller applications.

Internet of Things (IoT)

The Internet of Things (IoT) is a concept that connects various electronic devices through the internet network to enable data exchange and remote control. Arduino, as an open-source microcontroller, is widely used in IoT system development due to its ease of programming and support for various modules such as the ESP8266 and Ethernet Shield [10].



Source: telkomuniversity.ac.id

Fig. 1. IoT Concept

Sensor Transmitter

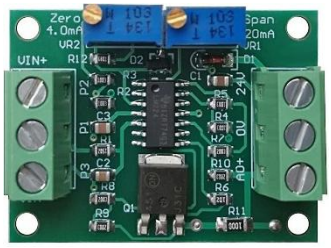
A sensor transmitter is a device that combines a sensor and a transmitter to detect, measure, and send data to monitoring or control systems in the form of analog signals (4–20 mA), digital signals (RS-485, Modbus), or wireless communication (Wi-Fi, Bluetooth) [11].

Source: https://www.emerson.com

Fig. 2. Transmitter Sensor

HW-685

The HW-685 current converter module is an input device that measures current in the 4–20 mA range and converts it into a voltage readable by other systems. This module is commonly used to detect process variables, such as temperature or pressure, via the 4–20 mA current signal [12].



Source: http://repository.unas.ac.id/id/eprint/9428

Fig. 3. HW-685

Arduino Uno R3

Arduino is a microcontroller platform based on the ATmega328 and is available in various more advanced variants. This platform is suitable for building minimalist yet reliable and fast electronic systems [5].

Source: https://store.arduino.cc/products/arduino-uno-rev3

Fig. 4. Arduino Uno R3

Arduino IDE

An Integrated Development Environment (IDE) is an integrated software environment used for application development. In the context of Arduino, the IDE is used to program various available functions through programming syntax. The Arduino IDE is built using the Java programming language and is equipped with C/C++ libraries known as Wiring, which simplify input and output operations [5].

Source: https://docs.arduino.cc/software

Fig. 5. Arduino IDE

ADS1115

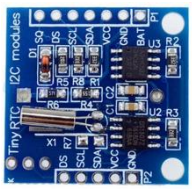
The ADS1115 is a low-power, 16-bit precision analog-to-digital converter (ADC) compatible with I2C, offered in a very small X2QFN-10 no-lead package and a VSSOP-10 package. The ADS1115 device integrates a low-drift voltage reference and oscillator. It also includes a programmable gain amplifier (PGA) and a digital comparator [13].

Source:https://jurnal.umj.ac.id/index.php/semnastek/article/view/1905/1558

Fig. 6. ADS1115

RTC DS1307

Powered by a 32.768 kHz crystal, the Maxim DS1307 is a popular 8-pin I2C-based RTC chip available in the market. Featuring automatic power-fail detection and a battery-switched circuit, the DS1307 maintains accurate operation even when the main power supply is lost. Its ultra-low power consumption, less than 500 nA on a 3-volt lithium backup battery (48 mAh or higher), enables the DS1307 to operate for over 10 years without main power [14].



Source:https://proceeding.pancabudi.ac.id/index.php/ICDSET/article/view/252/213

Fig. 7. RTC DS1307

Modul Micro SD Card

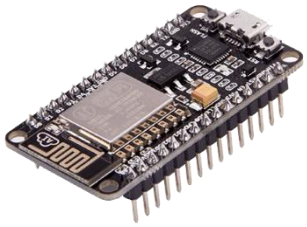
The microSD module is a card reader device equipped with a microcontroller to manage the read-write process on the microSD card. This module uses an SPI interface consisting of SS (chip select), MOSI (Master Out Slave In), MISO (Master in Slave Out), and SCK (Serial Clock) pins [15].

Source:https://ejurnal.itenas.ac.id/index.php/elkomika/article/view/2033/1842

Fig. 8. Micro SD Card Module

NodeMCU ESP8266

The NodeMCU ESP8266 is a microcontroller module that integrates the ESP8266 chip to provide Wi-Fi connectivity capabilities. The ESP8266 enables the microcontroller to connect to Wi-Fi networks, facilitating data communication between devices and the internet [16], [7]. The NodeMCU ESP8266 is selected due to its ease of programming, sufficient I/O pins, and ability to access the internet via Wi-Fi connection for data transmission and reception.

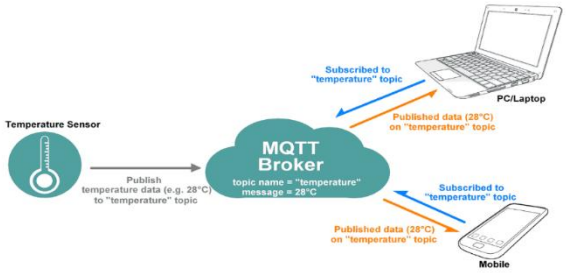


Source:https://www.ejournal.stmiklombok.ac.id/index.php/jire/article/view/381/166

Fig. 9. NodeMCU ESP8266

MQTT (Message Queue Telemetry Transport)

MQTT (Message Queue Telemetry Transport) is a lightweight publish-subscribe protocol operating over TCP/IP. MQTT is open, simple, and designed for easy implementation. The publish-subscribe messaging pattern requires a message broker, which is responsible for distributing messages to interested clients based on the message topic[3],[8].



Source:https://elmechtechnology.com/blog/penjelasan-detail-protokol-mqtt-message-queuing-telemetry-transport

Fig. 10. MQTT Architecture

MQTT Broker

The Adafruit IO platform provides a stable and lightweight MQTT broker service, supporting two-way communication between IoT devices and servers with token-based authentication and flexible topic structures. Adafruit IO has been widely used as an MQTT broker in various IoT applications due to its efficient data communication capabilities and the availability of an easy-to-configure web-based visualization interface [17].

III. METHODS

This study follows a systematic research methodology designed to obtain accurate and relevant data while ensuring that the research objectives are met. The methodology consists of the following stages:

1. Problem Identification: This initial stage involves understanding the challenges encountered in geothermal well monitoring, particularly related to data acquisition and real-time monitoring accuracy.
2. Literature Review: Relevant literature and prior studies are reviewed to gather comprehensive information and to build a theoretical framework supporting the development of the monitoring system.
3. System Requirement Analysis: Based on the problem identification and literature review, technical and functional requirements of the real-time monitoring system are analyzed and specified to guide the system development.
4. System Design: The system architecture is designed by selecting appropriate hardware components (such as sensors, microcontrollers, communication modules) and developing the software framework necessary for data acquisition, storage, transmission, and visualization.
5. Testing and Validation: The developed system undergoes laboratory-scale testing using a 4–20 mA signal generator to simulate sensor outputs. The testing evaluates the accuracy of readings, data transmission stability, and system responsiveness.
6. Data Analysis: Collected data from tests are processed and analyzed using analytical software to assess system performance, identify deviations, and verify reliability.
7. Reporting: The findings and outcomes from the research are documented comprehensively and prepared for dissemination in the form of scientific reports and publications.

This structured approach ensures that the research progresses logically from problem understanding to solution validation, providing reliable and reproducible results.

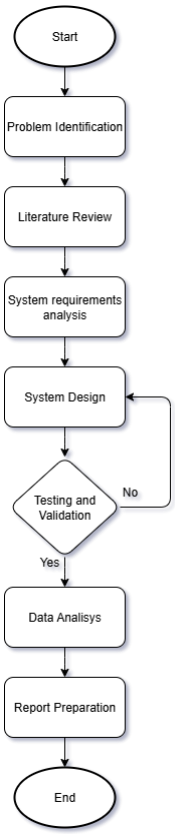


Fig. 11. Research Flow Chart

Hardware Requirement

The hardware required to implement a real-time geothermal well flow monitoring system using Internet of Things (IoT) technology consists of several components, including:

No.	*Hardware*	Fungsi
1.	Sensor	Output 4-20mA current signal
2.	HW-685	Converter current to voltage
3.	ADS1115	16-bit Analog to Digital Converter (ADC)
4.	Arduino Uno	Data processing unit
5.	ESP8266	Transmits data via network

Software Requirement

The software components required to operate and develop the geothermal well flow monitoring system include:

1. Arduino IDE
2. MQTT Adafruit IO
3. Visual Studio Code
4. HTML
5. PHP
6. JavaScript
7. Webhosting server

Overall System Block Diagram

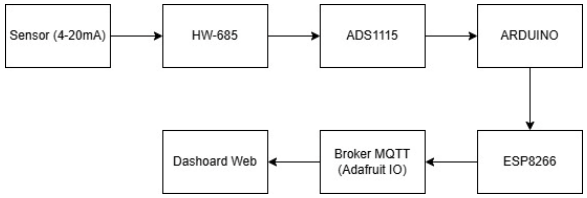


Fig. 12. Overall block diagram

The overall system consists of sensor transmitters that measure field parameters and output 4–20 mA signals. These signals are converted to voltage by the HW-685 module, then digitized by the ADS1115 ADC. The Arduino Uno processes the digital data, stores it locally, and sends it via ESP8266 Wi-Fi module using MQTT protocol to an MQTT broker (Adafruit IO). The broker distributes data to a web-based dashboard for real-time visualization and monitoring.

Processing and Storage Block Diagram

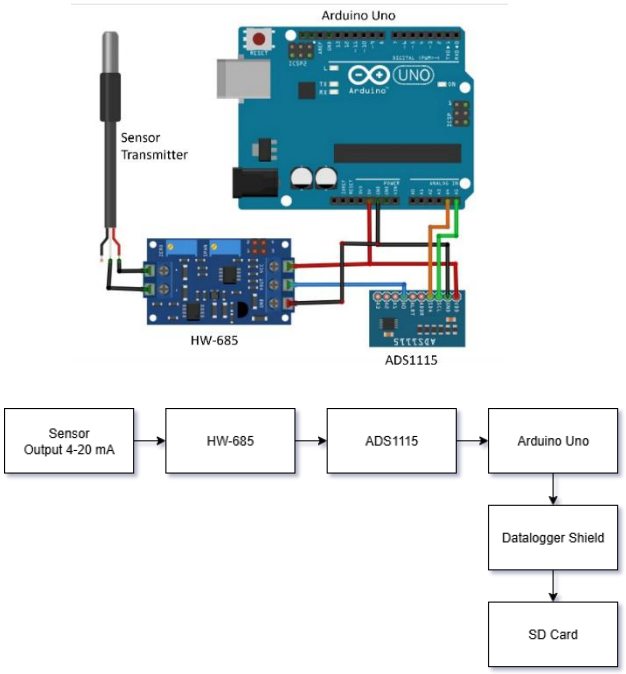


Fig. 13. Processing and storage block diagram

This block diagram illustrates the complete data flow from sensor acquisition to processing and local storage. Sensor outputs in the form of analog current (4–20 mA) are first converted to voltage using the HW-685 signal conditioning module, ensuring compatibility with subsequent analog-to-digital conversion. The ADS1115 ADC digitizes the conditioned signals with high resolution. Inside the Arduino microcontroller, the digital data is processed to convert raw readings into physical units representing pressure, temperature, and water flow rate. Additional calibration, filtering, and formatting routines are applied before storing the processed data on an SD card for local archival. This approach ensures data integrity, continuity, and accessibility, particularly during network outages or transmission failures, while enabling later retrieval for analysis.

Data Processing Flowchart on Arduino

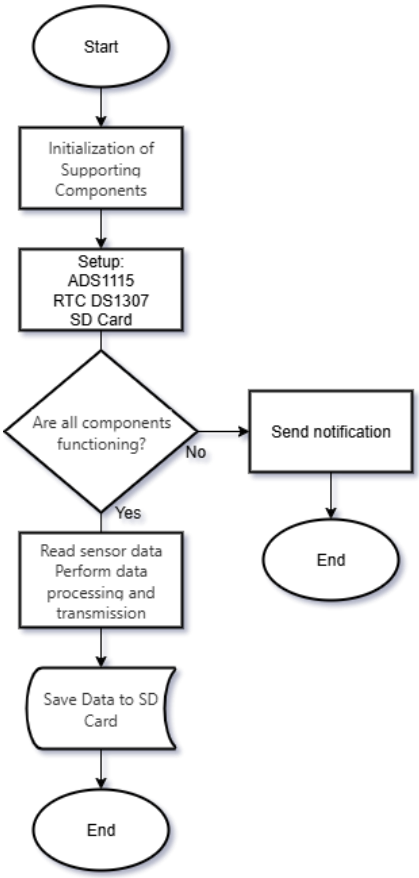


Fig. 14. Arduino flowchart processing

The Arduino Uno program was designed to manage sensor data acquisition, processing, and storage in a structured and reliable manner. Upon startup, the system initializes all hardware modules, including the ADS1115 analog-to-digital converter (ADC) for high-resolution signal digitization, the DS1307 real-time clock (RTC) for timestamping, and the SD Card module for local data storage. A verification routine is executed to ensure all modules are operational before proceeding. If verification fails, the Arduino issues a notification and terminates execution to prevent incomplete or corrupted data logging. If verification passes, the Arduino reads sensor data from connected transducers, processes the raw digital values into physical units representing pressure, temperature, and water flow rate, and stores the processed data on the SD Card.

In addition to autonomous logging, the Arduino receives control commands from the Downloader application, which can trigger downloading of stored data, adjust the recording interval, start or stop data recording, and query the current recording status. This integration allows both automated and user-controlled operation, ensuring flexibility and robustness in various field conditions.

The entire operational sequence is represented in the system flowchart (see Fig. 14), which serves as a visual reference for the program logic and hardware–software interaction. This structured workflow ensures reliable measurement, accurate physical unit conversion, and secure local data storage, even in the event of network interruptions.

Downloader Application Workflow

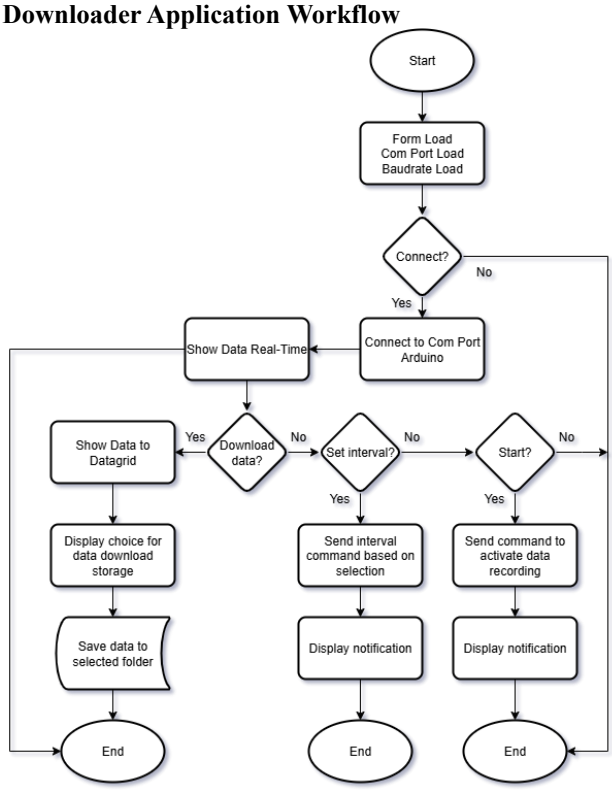


Fig. 15. Downloader flowchart

The Downloader application functions as the primary user interface for managing and retrieving data from the Arduino-based data logging system. Upon launch, the application establishes a communication link with the Arduino via a serial COM port connection. Once the connection is confirmed, the application can transmit specific commands to control the data logging process.

Core functionalities include downloading recorded data from the SD card to the local computer, configuring the data recording interval, starting and stopping the recording process, and requesting the current recording status. These commands are processed by the Arduino firmware in real time, allowing for immediate execution and confirmation of each operation. The application also incorporates error handling routines to detect communication failures or invalid responses from the Arduino, prompting the user with appropriate notifications to ensure reliable operation. This bidirectional interaction enables flexible field operation, as data can be retrieved, parameters adjusted, and logging sessions controlled without physical intervention at the hardware site.

The operational flow of the Downloader application is modeled in a system flowchart (see Fig. 15), highlighting its sequential logic, decision-making processes, and communication steps with the Arduino device. This design ensures efficient data management, enhances system usability, and maintains data integrity across the entire acquisition and storage workflow.

IV. Results And Discussion

The developed system was successfully assembled and tested to perform real-time monitoring of geothermal well flow test parameters, namely pressure, temperature, and water flow rate. The complete system integrates an Arduino Uno, NodeMCU ESP8266, ADS1115 ADC module, DS1307 RTC module, and an SD Card module, along with the necessary signal conditioning circuits and display units.

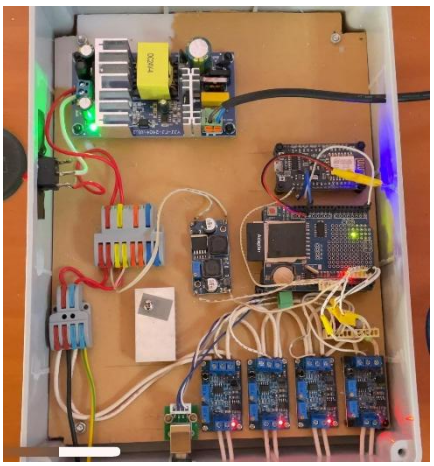


Fig. 16. System Design

Fig. 16. shows the physical realization of the developed real-time monitoring system for geothermal well flow testing. The above image presents the internal layout of the device, consisting of:

- DC step-down converter (24 V to 5 V) for powering the ESP8266 and Arduino.
- Arduino Uno microcontroller equipped with an SD Card module for reliable local data logging.
- RTC DS1307 module for accurate timekeeping during data acquisition.
- ESP8266 Wi-Fi module for wireless communication.
- ADS1115 ADC module for converting analog sensor outputs (4–20 mA signals) into digital data.
- DC Power supply 24 V.

Initial System Output in Serial Monitor

Upon system initialization, the microcontroller (Arduino Uno and NodeMCU ESP8266) begins acquiring sensor data from the connected measurement modules. This data includes wellhead pressure (WHP), low-pressure (LP), wellhead temperature (WHT), and brine flow rate. The raw sensor readings are immediately converted into physical units (Barg, °C, and m^3/h) before being displayed on the serial monitor.

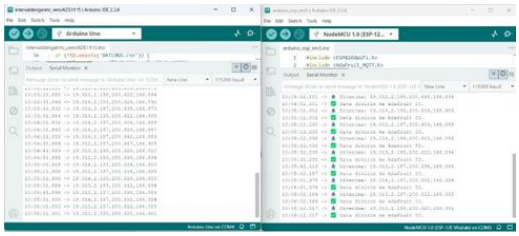


Fig. 17. Real-time output on serial monitor

Figure 17 shows the real-time output as observed in the Arduino IDE serial monitor for both devices. The Arduino Uno handles the initial data acquisition from the sensors, while the ESP8266 transmits the processed data to the MQTT broker. The display confirms that each parameter is successfully retrieved and formatted with its respective unit. The left panel in Figure 17. demonstrates the Arduino Uno's output, which primarily focuses on the direct readings from the data acquisition process. The right panel shows the ESP8266 output, indicating successful receipt and re-broadcast of the data. This stage is crucial in validating the proper functioning of the data acquisition and communication subsystems before logging or downloading operations are performed.

Data Logging via Downloader Application

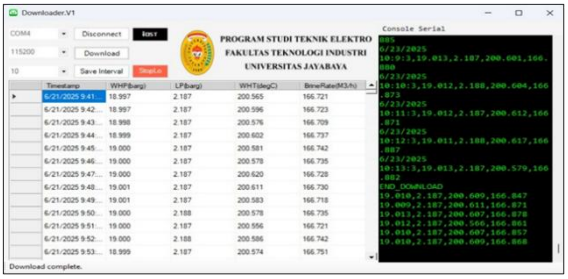


Fig. 18. Downloader Application

Fig. 16. Screenshot of the Downloader application interface used for managing data retrieval from the Arduino-based data acquisition system. The application establishes a serial connection with the Arduino to perform operations such as starting or stopping data recording, adjusting the logging interval, and downloading historical data stored on the SD card. The left panel serves as a downloaded data preview table, displaying sensor datasets (timestamp, WHP, LP, WHT, and brine flow rate) retrieved from the SD card before being saved to the local storage folder. The Arduino pre-processes these datasets by converting raw ADC readings into physical units, ensuring the table presents calibrated values. The right panel functions as a serial console monitor, showing real-time output from the Arduino, including current sensor readings, recording status updates, and operational messages such as download progress and completion notices. This two-panel layout enables users to validate downloaded datasets while simultaneously observing live system activity.

Local Storage Output

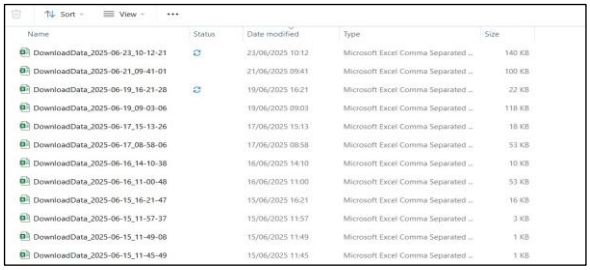


Fig. 19. File CSV on local storage

Upon completion of the download process via the Downloader application, measurement records from the Arduino's SD card are saved in Comma-Separated Values (CSV) format within the designated local directory, as depicted in Figure Z. Each CSV file is automatically named based on the download date and time (e.g., DownloadData_2025-06-21_19-21-27.csv), enabling chronological organization and easy retrieval.

This file-naming convention ensures traceability of datasets and facilitates version control during long-term monitoring. The CSV format allows compatibility with a wide range of data analysis tools, including Microsoft Excel, MATLAB, and Python-based analytics scripts. The file sizes vary depending on the duration and interval of data acquisition, ranging from a few kilobytes to over 100 kilobytes.

By employing this structured storage approach, the system guarantees that all acquired physical measurements such as wellhead pressure, low-pressure values, temperature, and brine rate are preserved in a universally readable format, ensuring accessibility for post-processing and archival purposes.

Real-Time Data Transmission via MQTT

After acquiring and processing sensor data into physical units (pressure, temperature, and water flow rate), the Arduino Uno transmits the formatted data to the ESP8266 module via serial communication. The ESP8266, configured as an MQTT client, establishes a connection with the designated MQTT broker using the TCP/IP protocol over Wi-Fi. Each set of physical measurement data is published to a specific topic, enabling real-time monitoring by any subscribed client application.

Fig. 17. shows the serial monitor output of the ESP8266 during system operation. The output confirms successful connection to the MQTT broker and the continuous publishing of sensor data in the desired physical units. The timestamp from the DS1307 RTC is included in each data payload, ensuring chronological accuracy during data reconstruction on the subscriber side.

The system design allows simultaneous data logging on the SD card and real-time data streaming over MQTT. This redundancy ensures data availability even if network connectivity is lost, as the local SD card retains the complete measurement history. Upon network restoration, real-time transmission resumes without manual intervention.

System Performance Evaluation

ADS1115	Gain_Twothirds	0 - 6.144 Volt																
	16 Bit	0 - 32767																
Sensor	Voltage (Volt)		ADC		WHP		WHT		LIP		Brine Rate		Persentase Error (%)					
4-20mA	Formula	Aktual	Formula	Aktual	Formula	Aktual	Formula	Aktual	Formula	Aktual	Formula	Aktual	ADC	WHP	WHT	LIP	Brine Rate	
4	1.00	1	5333	5315	0.00	0	0.00	0.00	0.00	0	0.00	0	0.34	0	0	0	0	
5	1.25	1.25	6666	6666	2.15	2.15	21.88	21.83	0.43	0.428	31.25	31.57	0.01	0.18	0.21	0.69	1.00	
6	1.50	1.5	8000	8001	4.31	4.31	43.75	43.74	0.86	0.872	62.50	62.816	0.02	0.06	0.03	1.16	0.50	
7	1.75	1.75	9333	9345	6.46	6.48	65.63	65.79	1.29	1.31	93.75	93.737	0.13	0.25	0.25	1.31	0.02	
8	2.00	2.01	10666	10740	8.62	8.63	87.50	87.96	1.72	1.748	125.00	125.73	0.69	0.16	0.52	1.39	0.58	
9	2.25	2.27	12000	12088	10.77	10.80	109.38	110.00	2.15	2.182	156.25	157.43	0.73	0.22	0.57	1.25	0.75	
10	2.50	2.51	13333	13383	12.93	12.90	131.25	132.06	2.59	2.602	187.50	188.78	0.37	0.23	0.61	0.63	0.68	
11	2.75	2.77	14666	14750	15.08	15.08	153.13	154.20	3.02	3.044	218.75	220.75	0.57	0.04	0.70	0.90	0.91	
12	3.00	3.02	16000	16102	17.24	17.25	175.00	176.23	3.45	3.481	250.00	252.31	0.64	0.05	0.70	0.97	0.92	
13	3.25	3.27	17333	17426	19.39	19.38	196.87	198.33	3.88	3.919	281.25	283.63	0.53	0.04	0.73	1.04	0.84	
14	3.50	3.53	18666	18812	21.55	21.56	218.75	220.33	4.31	4.355	312.50	315.69	0.78	0.05	0.72	1.05	1.01	
15	3.75	3.77	19999	20112	23.70	23.73	240.62	242.43	4.74	4.79	343.75	345.7	0.56	0.14	0.74	1.04	0.56	
16	4.00	4.03	21333	21489	25.86	25.92	262.50	264.42	5.17	5.232	375.00	377.81	0.73	0.24	0.73	1.17	0.74	
17	4.25	4.27	22666	22790	28.01	28.07	284.37	286.45	5.60	5.669	406.25	409.05	0.54	0.21	0.72	1.18	0.69	
18	4.50	4.53	23999	24135	30.16	30.21	306.25	308.44	6.03	6.105	437.50	441.17	0.56	0.14	0.71	1.18	0.83	
19	4.75	4.78	25333	25480	32.32	32.37	328.12	330.51	6.46	6.542	468.75	471.79	0.58	0.15	0.72	1.20	0.64	
20	5.00	5.03	26666	26814	34.47	34.529	350.00	352.27	6.89	6.968	500.00	503.24	0.55	0.16	0.65	1.05	0.64	

Fig. 20. Percentage error distribution

The developed system was evaluated in terms of measurement accuracy, data storage reliability, and network transmission stability. Since no calibrated industrial instruments were available, a 4–20 mA current generator was used as the reference source, with output values incrementally adjusted from 4 mA to 20 mA. The Arduino Uno processed these signals via the ADS1115 16-bit ADC module, and the resulting data were compared to values calculated using reference formulas in Microsoft Excel.

Fig. 20. presents the comparison between the calculated (Formula) and measured (Actual) values for voltage, ADC output, and all derived process parameters, including WHP, WHT, LIP, and Brine Rate. Across the full range of input currents, the percentage error for each parameter remained consistently below 1.5%, indicating a high degree of accuracy for the designed measurement system despite the absence of a precision-calibrated reference instrument.

Local storage performance was validated by inspecting CSV files saved on the SD card and subsequently downloaded via the Downloader Application (Fig. 18). All files were intact, with no evidence of corruption or incomplete logging, confirming stable write operations to the SD card during continuous acquisition.

Network performance was assessed via the ESP8266 MQTT log outputs, which showed consistent connectivity to the MQTT broker with negligible packet loss. In cases of temporary network interruptions, the system continued local logging and automatically resumed data transmission upon connection restoration.

These results demonstrate that the system is capable of accurate measurement, reliable local storage, and stable wireless transmission, ensuring robust performance in both standalone and networked operation modes.

V. Conclusion

Based on the design, implementation, and testing of the IoT-based real-time monitoring system for geothermal well flow tests, the following conclusions can be drawn:

1. The system successfully acquired analog signals from 4–20 mA sensors, converted them into digital data using the ADS1115 module, and processed them with the Arduino Uno.
2. Data were transmitted in real time to the MQTT broker (Adafruit IO) every 10 seconds via the ESP8266 and displayed visually on the dashboard UI in the form of numbers, graphs, and responsive gauges.
3. The local data recording function to the SD card operated automatically, storing data in .CSV format with a time-based structure to facilitate analysis.
4. The dashboard UI supported export to Google Sheets, allowing periodic (60-second interval) cloud storage with a structured format accessible from various devices.
5. The Downloader App was successfully implemented as a supporting application to control the recording process, display real-time data, and download data from the Arduino data logger.
6. The dashboard authentication system worked reliably, ensuring that only authorized users could access the data visualization.
7. Testing results indicated that all primary features functioned according to specifications, with sensor conversion deviations below 1.5%, and the system could handle connection disruptions (Wi-Fi, MQTT) without crashes or permanent data loss.

Overall, the system achieved the objective of providing flexible, IoT-based, real-time monitoring of geothermal well flow tests with both local and cloud storage capabilities.

References

[1] I. Yudiantoro and I. Nuraeni, "Evaluasi Potensi Produksi Sumur Panas Bumi Menggunakan Data Uji Komplesi Dan Data Produksi," *Jurnal Migasian*, vol. 2, no. 1, pp. 22–28, 2018, doi: 10.36601/jurnal-migasian.v2i1.20.

[2] N. M. Saptadji, "Teknik Panas Bumi," *Bandung, Penerbit ITB*, 2001.

[3] R. R. Pahlevi, P. Sukarno, and B. Erfianto, "Implementation of Event-Based Dynamic Authentication on MQTT Protocol," *Jurnal Rekayasa Elektrika*, vol. 15, no. 2, 2019, doi: 10.17529/jre.v15i2.13963.

[4] A. Y. Rangan, Amelia Yusnita, and Muhammad Awaludin, "Sistem Monitoring berbasis Internet of things pada Suhu dan Kelembaban Udara di Laboratorium Kimia XYZ," *Jurnal E-Komtek (Elektro-Komputer-Teknik)*, vol. 4, no. 2, pp. 168–183, Dec. 2020, doi: 10.37339/e-komtek.v4i2.404.

[5] M. I. Hafidhin, A. Saputra, Y. Rahmanto, and S. Samsugi, "Alat Penjemuran Ikan Asin Berbasis Mikrokontroler Arduino UNO," *Jurnal Teknik dan Sistem Komputer*, vol. 1, no. 2, pp. 59–66, 2020, doi: 10.33365/jtikom.v1i2.210.

[6] R. M. Antosia, "Voltmeter Design Based on ADS1115 and Arduino Uno for DC Resistivity Measurement," *JTERA (Jurnal Teknologi Rekayasa)*, vol. 5, no. 1, p. 73, Jul. 2020, doi: 10.31544/jtera.v5.i1.2019.73-80.

[7] A. P. Manullang, Y. Saragih, and R. Hidayat, "Implementasi Nodemcu Esp8266 Dalam Rancang Bangun Sistem Keamanan Sepeda Motor Berbasis Iot," *JIRE (Jurnal Informatika & Rekayasa Elektronika)* , vol. 4, no. 2, pp. 163–170, 2021, [Online]. Available: http://e-journal.stmiklombok.ac.id/index.php/jireISSN.2620-6900

[8] A. F. Oklilas, R. Zulfahmi, Ermatita, and A. P. Jaya, "Temperature Monitoring System Based on Protocol Message Queue Telemetry Transport (MQTT)," *Proceedings - 1st International Conference on Informatics, Multimedia, Cyber and Information System, ICIMCIS 2019*, pp. 61–66, 2019, doi: 10.1109/ICIMCIS48181.2019.8985356.

[9] A. Nurhadi and Ricki Sastra, "Pengembangan Aplikasi Web Pengajuan Cuti Pegawai Secara Online Studi Kasus: PT. Yapindo Transportama (PCP Express)," *Jurnal Bianglala Informatika*, vol. 5, no. 2, pp. 2338–9761, 2017, [Online]. Available: http://apache2tirad.net.

[10] A. R. Utamy, Siswanto, and Sutarti, "Prototype Wireless Sensor Network Sistem Pengukuran Debu Dan Suhu Udara Berbasis Mqtt Server," *PROSISKO: Jurnal Pengembangan Riset dan Observasi Sistem Komputer*, vol. 10, no. 2, pp. 152–164, 2023, doi: 10.30656/prosisko.v10i2.7158.

[11] B. Permana Ginting, D. Setiawan, and S. Yakub, "Implementasi IOT Pada Ruang Tangki Gas Menggunakan Pressure Transmitter Berbasis Node MCU," *Hal 109 JURNAL SISTEM KOMPUTER TGD*, vol. 3, no. 4, pp. 109–117, 2024, [Online]. Available: https://ojs.trigunadharma.ac.id/index.php/jskom

[12] A. Y. Mohammed and R. M. Hagem, "Low Cost Portable System for Converting Mosul Electrical Substations to Smart One's," 2021. [Online]. Available: https://rengj.mosuljournals.com

[13] W. Djatmiko, "Prototipe Resistansi Meter Digital," *Seminar Nasional Sains dan Teknologi*, no. November, pp. 1–8, 2017.

[14] S. RezaKhan, A. Kabir, and D. Ara Hossain, "Designing Smart Multipurpose Digital Clock using Real Time Clock (RTC) and PIC Microcontroller," *Int J Comput Appl*, vol. 41, no. 9, pp. 39–42, 2012, doi: 10.5120/5573-7668.

[15] N. Pranoto, "PROTOTIPE SISTEM PRESENSI KELAS BERBASIS RFID DAN MICRO SD Disusun sebagai salah satu syarat menyelesaikan Program Studi Strata I," 2017.

[16] A. D. Pangestu, F. Ardianto, and B. Alfaresi, "Sistem Monitoring Beban Listrik Berbasis Arduino Nodemcu Esp8266," *Jurnal Ampere*, vol. 4, no. 1, p. 187, 2019, doi: 10.31851/ampere.v4i1.2745.

[17] M. Syafaat, A. N. Ramadhan, R. B. Syafiun, and D. A. Haerunnisa, "IoT-BASED SMART GARDEN USING MQTT PROTOCOL WITH ADAFRUIT IO APP," *Jurnal Teknik Informatika (Jutif)*, vol. 4, no. 4, pp. 723–732, Aug. 2023, doi: 10.52436/1.jutif.2023.4.4.636.

[18] M. Php, C. S. S. D. A. N. Uml, C. Christian, and A. Voutama, "Implementasi Aplikasi Antrian Pencucian Mobil Berbasis Web," vol. 8, no. 2, pp. 2243–2248, 2024.

[19] M. DirgaF, "APLIKASI E-LEARNING SISWA SMK BERBASIS WEB," 2021. [Online]. Available: https://jurnal.umpar.ac.id/index.php/sylog